

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

APPLICATION FOR LETTERS PATENT

**Single Window Navigation Methods and Systems**

Inventor(s):

Scott Ruthfield

Richard Wolf

Satoshi Nakajima

ATTORNEY'S DOCKET NO. MS1-560US

007200" 05205560

1 **RELATED APPLICATIONS**

2 The following patent applications are related to the present application, are  
3 assigned to the assignee of this patent application, and are expressly incorporated  
4 by reference herein:

- 5
- 6 • U.S. Patent Application Serial No. \_\_\_\_\_, entitled "Methods  
7 and Systems of Providing Information to Computer Users", bearing  
8 attorney docket number MS1-557us, and filed on the same date as  
9 this patent application;
  - 10 • U.S. Patent Application Serial No. \_\_\_\_\_, entitled "Methods,  
11 Systems, Architectures and Data Structures For Delivering Software  
12 via a Network", bearing attorney docket number MS1-559us, and  
13 filed on the same date as this patent application;
  - 14 • U.S. Patent Application Serial No. \_\_\_\_\_, entitled "Network-  
15 based Software Extensions", bearing attorney docket number MS1-  
16 563us, and filed on the same date as this patent application;
  - 17 • U.S. Patent Application Serial No. \_\_\_\_\_, entitled  
18 "Authoring Arbitrary XML Documents using DHTML and XSLT",  
19 bearing attorney docket number MS1-583us, and filed on the same  
20 date as this patent application;
  - 21 • U.S. Patent Application Serial No. \_\_\_\_\_, entitled  
22 "Architectures For And Methods Of Providing Network-based  
23 Software Extensions", bearing attorney docket number MS1-586us,  
24 and filed on the same date as this patent application.
  - 25 • U.S. Patent Application Serial No. \_\_\_\_\_, entitled "Task  
Sensitive Methods And Systems For Displaying Command Sets",  
bearing attorney docket number MS1-562us, and filed on the same  
date as this patent application.

20 **TECHNICAL FIELD**

21 This invention relates generally to computing systems and methods, and  
22 more particularly concerns systems and methods that facilitate creation and use of  
23 information within a computing environment.  
24  
25

[illegible]

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10
- 11
- 12
- 13
- 14
- 15
- 16
- 17
- 18
- 19
- 20
- 21
- 22
- 23
- 24
- 25

15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

1 message, the message is likely to be presented in yet another window. If the user  
2 chooses to respond to the email message, they typically prepare their response in a  
3 "reply" window which is yet a fourth window that the user must manage.

4 The above-described scenario constitutes a simple example of a window  
5 management scenario when the user has opened only two applications. Consider  
6 the case where a user has multiple applications (e.g. four or more) that they are  
7 working in throughout the day. As a specific example, consider the following  
8 four exemplary applications that a user might find necessary to use during the  
9 course of their computing day: a word processing application, a presentation  
10 application (such as Microsoft Powerpoint), a web browser application (such as  
11 Microsoft Internet Explorer), and an email application.

12 What many users typically do is they open all of the applications and then  
13 manage each application's separate and different window as they need access to a  
14 different application. The application managing part of the screen uses what is  
15 referred to as a "windowing environment." Yet, windowing environments are not  
16 necessarily intuitive to all potential computer users. User studies have consistently  
17 shown that one of the biggest hurdles for new users of a windowing environment  
18 is learning to understand a windowing system, where windows can "layer" on top  
19 of each other. Consider, for example, Fig. 1 which shows an exemplary user  
20 display 10 that includes four exemplary windows 12, 14, 16, 18, respectively  
21 dedicated to a word processor application, a presentation application, a web  
22 browser application, and an email application. These windows are layered on top  
23 of one another which, for a new computer user, can be a difficult concept to  
24 understand and manage. For example, some users might not understand how to  
25 use a task bar to manage the windows. They might, for example, inadvertently

close an application when they simply intended to minimize it. Additionally, the users might not appreciate or understand how to move the separate windows around on their display. Further, different windows can sometimes be inadvertently clicked by a user. For example, while a user is in a window associated with one application, they may inadvertently click the edge of a window associated with another application, whereupon they find themselves in the middle of a different application. Finally, many users simply do not comprehend that their screen has multiple layers: they only think of the top window as the one that they can interact with, as if the previous ones were lost.

Windowing environments, however, do not just pose challenges to newer computer users: they can sometimes pose challenges to users who are familiar with such environments. Specifically, management of multiple windows can be distracting to computer users, particularly when a user has many different windows that they are attempting to manage. For example, if a user has many different applications that they are executing that are being managed by a task bar at the bottom of the user's display, to switch from one application to another, the user must find the appropriate task bar portion that references an application of interest. The user must then click on the task bar portion to pull up the application. This scenario can be complicated if, for example, the user has multiple documents in one application that have been minimized, e.g. multiple word processing documents or multiple email messages that they might intend to respond to during the course of their day. This complicates the scenario because now the task bar must maintain an entry for not only each of the user's applications, but each of the documents within each application that might have been minimized by the user as well. At this point, accessing the minimized

1 applications or document is not an easy task, but rather has devolved into a trial  
2 and error hunting process. This is not an efficient way for busy users to manage  
3 their applications and documents.

4 Accordingly, this invention arose out of concerns associated with  
5 improving the efficiencies with which a computer user interacts with their  
6 computer.

### 7 8 SUMMARY

9 In one embodiment, a single navigable window is provided and can be  
10 navigated by a user to and between different functionalities. A functionality is  
11 analogous to an application program. The different functionalities enable a user to  
12 accomplish different tasks, e.g. word processing tasks, email tasks, calendar tasks  
13 and the like. The single navigable window and the functionalities to which it can  
14 be navigated are advantageously provided by a single application that, in turn,  
15 provides a very high degree of integration between the functionalities.

16 In the described embodiment, a user is presented with a user interface (UI)  
17 that contains both the single window and navigation instrumentalities that allow  
18 the user to navigate between the different functionalities. Exemplary navigation  
19 instrumentalities include links associated with each functionality that can be  
20 clicked on by a user to access a particular functionality. As the user clicks on  
21 various links and engages in different activities within the functionalities, a  
22 navigation model maintains a "travel log" of where the user has been so that the  
23 user can return to various functionalities to complete tasks or initiate new tasks. In  
24 one implementation the travel log is implemented in the form of a navigation stack  
25 that utilizes a "back and truncate" model. Advantageously, another of the



1 accounted for so that the user's navigation activities bear a logical correspondence  
2 with their actions.

3 In the document-centric embodiment, the user can author different types of  
4 documents by navigating the single window to a blank document of a particular  
5 type, and then authoring the document in an appropriate manner. The authored  
6 document can then be automatically published by the system in the appropriate  
7 manner (e.g. saved to a particular collection in the event of a free-form document,  
8 transmitted to a recipient in the event of an email message etc.). One particularly  
9 advantageous feature of the document-centric single navigable window is that it  
10 natively understands and can view different types of document collections. By  
11 navigating the window to a particular document collection, the user can select  
12 individual documents of the corresponding type and operate upon them in some  
13 manner.

14 In one embodiment, an extensible software platform is provided and serves  
15 as a basis to incorporate or integrate different "functionalities" into the single  
16 application program that provides the single navigable window. That is, both  
17 individual functionalities and the main application itself are extensible.

18 In one particularly advantageous embodiment, access to functionalities in  
19 accordance with the single navigable window application is provided via a  
20 network such as the Internet. This provides one basis for a subscriber model  
21 service in which the functionalities can be packaged and offered for sale to various  
22 subscribers.



## **BRIEF DESCRIPTION OF THE DRAWINGS**

Fig. 1 is an exemplary display offered by a windowing environment in accordance with the prior art.

Fig. 2 is a high level block diagram of an exemplary computing system that is suitable for implementing the described embodiments.

Fig. 3 is an exemplary user interface (UI) in accordance with one described embodiment.

Fig. 4 is an exemplary specific user interface that is displaying one exemplary document-centric functionality in accordance with an example illustrating the described embodiment.

Fig. 5 is an exemplary specific user interface that is displaying one exemplary document-centric functionality in accordance with an example illustrating the described embodiment.

Fig. 6 is a flow diagram that describes steps in a method in accordance with the described embodiment.

Fig. 7 is a block diagram that illustrates an exemplary navigation model in the form of a navigation stack that implements a "back-and-truncate" model.

Fig. 8 is a block diagram that illustrates an exemplary navigation model in the form of a navigation stack, and visually demonstrates one inventive navigation stack manipulation in accordance with the described embodiment.

Fig. 9 is a flow diagram that describes steps in a method in accordance with the described embodiment.

Fig. 10 is a flow diagram that describes steps in a document-centric method in accordance with the described embodiment.

1 Fig. 11 is a flow diagram that describes steps in a document-centric method  
2 in accordance with the described embodiment.

3 Fig. 12 is a high level block diagram that illustrates concepts in accordance  
4 with one embodiment.

5 Fig. 13 is a high level block diagram that illustrates concepts in accordance  
6 with one embodiment.

## 7 8 **DETAILED DESCRIPTION**

### 9 **Overview**

10 The embodiments described just below provide improved methods and  
11 systems for creating and using information in a computing environment. The  
12 inventive methods and systems address, among other things, user issues relating to  
13 computing in a multi-window environment as discussed above.

14 The inventive methods and systems provide a single navigable window that  
15 can be used by a user to navigate to and between multiple different functionalities  
16 that are provided by a single application program. By "single navigable window"  
17 is meant that users can move from one functionality to another, and within  
18 functionalities, all within one window, and then can move back and forth between  
19 previously-visited functionalities in a session in a prescribed order based on the  
20 order they visited those places. The functionalities enable the user to complete  
21 different tasks, and the single window enables the user to navigate between  
22 functionalities, and hence tasks, in a seamless manner. By having only one  
23 window, the user is relieved of the duties of managing multiple windows. By  
24 having all of the functionalities presented within a single application, the user is  
25 provided with a highly integrated software product that greatly improves the user's

1 computing experience. Exemplary functionalities are described in the context of  
2 document-centric functionalities such as word processing and email  
3 functionalities.

4 The inventive methods and systems make novel use of a navigation model  
5 that manages the user's navigation activities to and between the different  
6 functionalities. The described navigation model is a navigation stack that  
7 conforms generally to a "back-and-truncate" model.

8 Navigation instrumentalities are provided and enable the user to navigate  
9 among the different functionalities. In the described embodiment these navigation  
10 instrumentalities include, without limitation, links to each of the different  
11 functionalities as well as browser-like navigation buttons.

12 Context-sensitive command sets can also be provided in a user interface  
13 along with the single navigable window. The context sensitive command sets  
14 include commands that automatically change as the user's computing context  
15 changes, e.g. as the user moves from functionality to functionality.

16 In one embodiment, the single application is defined as a software platform  
17 that is extensible to receive and incorporate different functionalities. The  
18 functionalities can be provided as software modules that can be sent over a  
19 network such as the Internet. The extensible software platform provides a basis to  
20 offer a subscriber or fee-based service where different subscribers can, for a fee,  
21 access different functionalities via a network such as the Internet.

### 22 23 Exemplary Computer System

24 Fig. 2 shows an exemplary computer system that can be used to implement  
25 the embodiments described herein. Computer 130 includes one or more

1 processors or processing units 132, a system memory 134, and a bus 136 that  
2 couples various system components including the system memory 134 to  
3 processors 132. The bus 136 represents one or more of any of several types of bus  
4 structures, including a memory bus or memory controller, a peripheral bus, an  
5 accelerated graphics port, and a processor or local bus using any of a variety of  
6 bus architectures. The system memory 134 includes read only memory (ROM)  
7 138 and random access memory (RAM) 140. A basic input/output system (BIOS)  
8 142, containing the basic routines that help to transfer information between  
9 elements within computer 130, such as during start-up, is stored in ROM 138.

10 Computer 130 further includes a hard disk drive 144 for reading from and  
11 writing to a hard disk (not shown), a magnetic disk drive 146 for reading from and  
12 writing to a removable magnetic disk 148, and an optical disk drive 150 for  
13 reading from or writing to a removable optical disk 152 such as a CD ROM or  
14 other optical media. The hard disk drive 144, magnetic disk drive 146, and optical  
15 disk drive 150 are connected to the bus 136 by an SCSI interface 154 or some  
16 other appropriate interface. The drives and their associated computer-readable  
17 media provide nonvolatile storage of computer-readable instructions, data  
18 structures, program modules and other data for computer 130. Although the  
19 exemplary environment described herein employs a hard disk, a removable  
20 magnetic disk 148 and a removable optical disk 152, it should be appreciated by  
21 those skilled in the art that other types of computer-readable media which can  
22 store data that is accessible by a computer, such as magnetic cassettes, flash  
23 memory cards, digital video disks, random access memories (RAMs), read only  
24 memories (ROMs), and the like, may also be used in the exemplary operating  
25 environment.

1 A number of program modules may be stored on the hard disk 144,  
2 magnetic disk 148, optical disk 152, ROM 138, or RAM 140, including an  
3 operating system 158, one or more application programs 160, other program  
4 modules 162, and program data 164. A user may enter commands and  
5 information into computer 130 through input devices such as a keyboard 166 and a  
6 pointing device 168. Other input devices (not shown) may include a microphone,  
7 joystick, game pad, satellite dish, scanner, or the like. These and other input  
8 devices are connected to the processing unit 132 through an interface 170 that is  
9 coupled to the bus 136. A monitor 172 or other type of display device is also  
10 connected to the bus 136 via an interface, such as a video adapter 174. In addition  
11 to the monitor, personal computers typically include other peripheral output  
12 devices (not shown) such as speakers and printers.

13 Computer 130 commonly operates in a networked environment using  
14 logical connections to one or more remote computers, such as a remote computer  
15 176. The remote computer 176 may be another personal computer, a server, a  
16 router, a network PC, a peer device or other common network node, and typically  
17 includes many or all of the elements described above relative to computer 130,  
18 although only a memory storage device 178 has been illustrated in Fig. 2. The  
19 logical connections depicted in Fig. 2 include a local area network (LAN) 180 and  
20 a wide area network (WAN) 182. Such networking environments are  
21 commonplace in offices, enterprise-wide computer networks, intranets, and the  
22 Internet.

23 When used in a LAN networking environment, computer 130 is connected  
24 to the local network 180 through a network interface or adapter 184. When used  
25 in a WAN networking environment, computer 130 typically includes a modem 186



## Single Navigable Window

In accordance with one embodiment, software provides a user interface (UI) that presents a user with a single navigable window that can be navigated from functionality to functionality and inside individual functionalities by a user. The user interface enables the user to effectively manage multiple windows, and hence multiple functionalities, by presenting only one window at a time. This is different from the traditional windowing environment because the windows that pertain to the individual functionalities are not layered on one another and do not need separate management. Another noteworthy way that the single navigable window varies from the traditional windowing environment is that the various functionalities are provided by a single application. That is, in the traditional windowing environment, it is very typical for multiple windows to be provided by multiple different applications that are opened by a user, e.g. a word processing application will have one window, an email application will have another window, a web browsing application will have another window, and the like. All of these windows are separate and require separate management by the user. In the present case, various functionalities that were once the domain of individual separate applications are now the domain of a single integrated application which provides its own window management scheme. The window management scheme is embodied in the form of a single navigable window that can be navigated by a user from functionality to functionality.

A user, through the use of navigation instrumentalities can navigate between the functionalities and when doing so, the single window ensures that only one of these functionalities is presented to a user at a time. In the described embodiment, one navigation instrumentality is provided in the form of a web

1 browser-like navigation tool. The choice of a web browser-like navigation tool  
2 follows from concerns that navigation instrumentalities be of a type that is readily  
3 understood by most individuals familiar with computing environments. Thus,  
4 when a user first encounters the inventive navigable single window concept for the  
5 first time, they do not have to learn an unfamiliar navigation concept. Another  
6 navigation instrumentality includes links to each of the multiple different  
7 functionalities. These links can be clicked on by a user and the single navigable  
8 window is automatically navigated to the selected functionality.

9 Fig. 3 shows but one exemplary user interface (UI) 300 in accordance with  
10 one described embodiment. It will be appreciated that other UIs could be used to  
11 implement the inventive concepts described herein and that the illustrated UI  
12 constitutes but one way of doing so. In the illustrated example, UI 300 includes a  
13 navigation bar 302, one or more command areas 304, and a display or document  
14 area 306 that constitutes the single navigable window.

15 Navigation bar 302 is located adjacent the top of display area 306 and  
16 contains browser-like navigation buttons 308 in the form of a "backward" button,  
17 a "forward" button, a "stop" button and the like. The navigation bar can be  
18 located anywhere on the UI. Its illustrated placement, however, is similar in  
19 appearance to the placement of traditional web browsing navigation features. In  
20 addition to the navigation buttons 308, the navigation bar 302 also includes links  
21 310 to the different functionalities that can be accessed by the user. In the  
22 illustrated example, links to three exemplary functionalities (i.e. functionality 1,  
23 functionality 2, and functionality 3) are shown. These functionalities are typically  
24 different functionalities that can enable a user to complete different respective  
25 tasks. Examples of different tasks are given below in more detail. These





1 the world icon). These functionalities are so-called "document-centric"  
2 functionalities because they all relate in some way to a document that a user  
3 interacts with, e.g. a Web page document, an email document, a calendar  
4 document, etc.

5 Fig. 4 shows an example of a display that is rendered in the display area  
6 306 when a user clicks on the link to the browser functionality. By clicking on the  
7 link (i.e. the home icon) to the browser functionality, single application program  
8 software executing on the user's computer executes to implement a browser  
9 functionality. In this example, the browser functionality displays the user's home  
10 page in display area 306. Notice also that navigation buttons 308 are provided for  
11 navigation between the different selectable functionalities. The command areas  
12 304 contain command sets that include commands that are specific to the context  
13 that the user has selected. In this example, the user's context is a browsing  
14 context. Accordingly, the leftmost command area contains commands that are  
15 specific to the browsing functionality. Such commands include ones that a user  
16 would normally expect to find in a web browser. Notice also that the command  
17 area 304 adjacent the top of display area 306 also contains commands that are  
18 specific to the browsing context, i.e. "Add to Favorites" and an address well in  
19 which the user can type a URL of a particular destination web site.

20 Fig. 5 shows an example of a display that is rendered in the display area  
21 306 when the user clicks on the link to the mail functionality (i.e. the folder icon).  
22 By clicking on this link, single application program software executing on the  
23 user's computer executes to implement the mail functionality. In this example, the  
24 mail functionality displays a user's in box with messages that have been received  
25 by the user. Notice that the leftmost command area has been minimized by the

1 user and that the command area adjacent the top of the display area 306 contains  
2 commands that are specific to the user's current context, e.g. "New" for generating  
3 a new email message, "Reply" for replying to an email message, "Reply to All"  
4 for replying to all recipients of an email message and the like.

5 Likewise, although not specifically illustrated, the user could have displays  
6 for the planner, contacts, documents, and links functionalities presented in the  
7 display area 306 by simply clicking on the links to these specific functionalities.  
8 The navigation bar 308 provides the user with the ability to navigate through these  
9 different functionalities in a browser-like manner.

10 It is important to note that the above example constitutes but one exemplary  
11 way in which multiple different functionalities can be presented to a user within  
12 the construct of a navigable structure. It should be understood that the specifically  
13 illustrated functionalities (i.e. browser, mail, planner etc.) constitute specific  
14 examples of different functionalities that are capable of being incorporated into the  
15 single application program that provides the navigable window. Accordingly,  
16 other different functionalities can be employed. This aspect is discussed in more  
17 detail in the section entitled "Extensible Functionalities" below.

18 Fig. 6 is a flow diagram that describes steps in a method in accordance with  
19 the described embodiment. The illustrated method can be implemented in any  
20 suitable hardware, software, firmware, or combination thereof. In the illustrated  
21 example, the method is implemented in software.

22 Step 600 provides a single application program with multiple different  
23 functionalities. The functionalities, as pointed out above, are advantageously  
24 different so as to enable a user to accomplish different tasks. One specific non-  
25 limiting example of different functionalities was given above in the context of

document-centric functionalities that enable a user to make use of browser, mail, planner, contacts, documents, and links functionalities. Step 600 can be implemented by configuring a computing device, such as a user's computer, with the single application program having the multiple different functionalities. As will be seen in various sections below, this step can also be implemented by providing a software platform in the form of a generic single application shell that is extensible and adaptable to receive different extensions or software modules that embody various different functionalities. These different extensions are then presented to the user in the context of the single application having the multiple different functionalities.

These extensions, as described below, can be delivered to the platform in any suitable way and through any suitable delivery mechanism. For example, one way of delivering the various extensions or functionalities is to deliver them via a network such as an Intranet or the Internet. Regardless of the manner in which the single application is provided, step 602 presents a user interface (UI) with a single window and links to the multiple different functionalities. The UI can also advantageously include navigation instrumentalities that enable a user to navigate between the different functionalities in a browser-like manner. Figs. 3-5 give specific examples of an exemplary UI that can be used in accordance with the described embodiment. Step 604 ascertains whether a user has selected a particular link to a functionality or whether the user has used one of the navigation instrumentalities to navigate to a particular functionality. If a user has not done either, the method branches back to step 602. If, on the other hand, a user has selected a particular link or used a navigation tool to navigate to a particular functionality, step 606 presents a functionality-specific display within the single

1 window. That is, the single navigable window is navigated by the software to the  
2 selected functionality. Specific examples of this were given above in connection  
3 with Figs. 4 and 5 in which browsing and mail functionalities were respectively  
4 displayed within display area 306. In connection with presenting the  
5 functionality-specific display in step 606, step 608 can present functionality-  
6 specific commands in a command area of the UI. This is advantageously done  
7 automatically as a user navigates from functionality to functionality. That is, as a  
8 user changes functionalities, command sets that are specific to the user's current  
9 context or functionality are automatically displayed in the command area. Step  
10 608 then branches back to step 604 to ascertain whether the user has navigated to  
11 another functionality.

12 Hence, in the above example, a single application comprises multiple  
13 functionalities that enable a user to accomplish different tasks. These multiple  
14 functionalities are presented in the context of a single window that is navigable by  
15 a user between the different functionalities. Advantageously, navigation  
16 instrumentalities are provided that are, in some instances, browser-like in  
17 appearance (although not necessarily in behavior, as will be discussed below) and  
18 allow the user to navigate between the application-provided functionalities in a  
19 browser-like manner. Functionality-specific commands can be automatically  
20 presented to the user when they navigate to a particular functionality.

### 21 22 **Navigation model**

23 In the described embodiment, a navigation model is utilized to manage a  
24 user's navigation activities within the single application that provides the multiple  
25 different functionalities. Although any suitable navigation model can be used, in

0621001005-MSI-560-PAT-APP.DOC

1 the described embodiment a so-called "back-and-truncate" navigation stack is  
2 used. The basic concept of a back-and-truncate model is known and forms the  
3 basis for many different web browsers on the market today. Essentially, the back-  
4 and-truncate model makes use of a navigation stack that is truncated when the user  
5 navigates back  $n$  times and then forward to a new document.

6 Consider, for example, Fig. 7 which illustrates how an exemplary  
7 navigation stack works in connection with the described embodiment. Essentially,  
8 when a user is presented with the single application UI, they can select links from  
9 different areas. First, they might select a push button link (e.g. 310 in Fig. 3) from  
10 the navigation bar; second, they might use the forward and back buttons 308;  
11 third, they might select a link that is display in the display area 306 (e.g. when in  
12 the browsing functionality, there may be links to different web pages that the user  
13 can select). As the user makes their way about the various functionalities and  
14 selectable links, a navigation stack is built and maintained. In the described  
15 embodiment, the navigation stack is maintained in memory, but could easily be  
16 maintained in a store to preserve the user's navigation stack for future sessions.

17 In the Fig. 7 example, a first navigation stack 700 can be established as  
18 follows: a user initiates the single window application and is presented with a UI  
19 such as the one shown in Fig. 3. From there, a user may click on a link to the first  
20 functionality which establishes a first entry 702 in the navigation stack. While  
21 engaging the first functionality the user may take part in a certain activity which  
22 results in a new display being presented in the display area 306 (Fig. 3).  
23 Accordingly, a second entry 704 appears in the navigation stack 700. As the user  
24 navigates from functionality to functionality (and takes part in activities within the  
25 functionalities) the navigation stack grows and shrinks in accordance with the

exemplary employed back-and-truncate model. This might result in the illustrated entries 706, 708, 710. Assume now that the user employs the “back” button in the navigation bar to move back through the navigation stack. This action is illustrated by arrow A in Fig. 7. When the user arrives at entry 704 in the navigation stack, they click on a different link that navigates their window to functionality 4 (entry 712 in the navigation stack). At this point, the user has moved back through the navigation stack and the stack has been truncated to include only entries 702, 704, and 712. From functionality 4, the user takes part in an activity that presents a display that results in entry 714 in the navigation stack. The above example essentially illustrates the functionality of a back-and-truncate navigation model. In the inventive systems and methods, improvements on this model are made to ensure that the user’s navigation experience is logically consistent with actions that the user can engage in.

As an example, the inventive systems and methods provide for navigation stack operations and manipulations that are not found in the typical contemporary navigation models. Specifically, in the present case, the navigation stack can be manipulated to delete, add, and modify navigation stack entries to ensure that consistency is maintained. Consider the following elementary example. Assume that the single window navigation system is employed in the context of functionalities that include the ones described above in connection with Figs. 4 and 5. Assume also that a user navigates from their home page to the email functionality and clicks the “new” button to author a new email message. At this point, the navigation stack will contain entries that look like those in Fig. 8 at 800, 802, and 804. Assume now that the user clicks the “send” button to send the email message to the recipient and then navigates to their planner functionality as

1 indicated by entry 806. Now assume that the user navigates backward through the  
2 navigation stack using their "back" button. If the navigation stack were not  
3 manipulated at all, then when the user navigated back to the new email message  
4 804 (which was sent and is physically and logically no longer present), the user  
5 would likely receive an error message because the document corresponding to that  
6 navigation stack entry is no longer available. Instead, the inventive systems and  
7 methods ensure that consistency is maintained by removing the entry 804 and  
8 reorganizing the navigation stack so that the entry preceding entry 806 is now  
9 entry 802.

10 Consider another example with reference to Fig. 8. Assume that a user  
11 navigates to their email functionality and reads a message that contains a browser  
12 link to a web page. Assume also that they click on the browser link which  
13 navigates their single window to the web page. The navigation stack will thus  
14 contain entries that look like those in Fig. 8 at 810, 812, and 814. Assume now  
15 that the user wishes to send an email message to the sender of their original  
16 message, but rather than navigating back through the navigation stack by using the  
17 "back" button, they simply click the mail link in their navigation bar and thus add  
18 entry 816 to the navigation stack. Assume now, while at the email functionality  
19 the user decides to delete the mail message that contained the browser link. When  
20 the user deletes the email message corresponding to entry 812, the software checks  
21 the navigation stack and removes the appropriate entry corresponding to the  
22 deleted message, i.e. entry 812. The software then reorganizes the navigation  
23 stack so that entry 810 leads to entry 814 and vice versa.

24 The above examples illustrate an important characteristic of some of the  
25 inventive navigation model manipulation operations that distinguish them from



09599999-06400

1 ordinary back-and-truncate operations. For example, here, manipulation of the  
2 navigation model, i.e. removing entries, adding entries and reorganizing entries  
3 takes place in response to user actions that are not necessarily navigation  
4 activities, i.e. the user moving to or between functionalities. Rather, the user's  
5 actions that result in the navigation stack manipulations in these examples  
6 constitute actions that are supported inherently by the various functionality to  
7 which the user has happened to navigate. These user actions necessarily impact  
8 the logical association of entries in the navigation model. Without the inventive  
9 navigation model manipulation operations, there is a high degree of likelihood that  
10 the single navigable window interface would present the user with a degraded  
11 experience that is logically inconsistent with the user's navigation activities and/or  
12 actions within a particular functionality, e.g. by presenting the user with a  
13 logically non-existent document when the user navigates backward in the  
14 illustrated navigation stack.

15 Fig. 9 is a flow diagram that describes steps in a navigation model  
16 management method in accordance with the described embodiment. The described  
17 method can be implemented in any suitable hardware, software, firmware or  
18 combination thereof. In the present example, the method is advantageously  
19 implemented in software in connection with a single application that contains  
20 multiple functionalities that can be navigated using a single navigable window as  
21 described above.

22 Step 900 establishes a navigation model responsive to user navigation  
23 activities. In the example given above, the navigation model is established in  
24 memory and manages the user's navigation activities as they navigate from  
25 functionality to functionality in the context of a single navigable window. In the



## Document centric Application

In the embodiment described in connection with Figs. 4 and 5, the functionalities that are available to the user relate to "document-centric" functionalities. Document-centric functionalities include such things as word processing functionalities, email functionalities, planner functionalities and the like—where the underlying object that the user is operating on is a document. Other document-centric functionalities can, of course, be provided and are not to be limited to those shown in the above examples. Such functionalities can include, without limitation, financial management functionalities, travel functionalities, medical functionalities, income tax functionalities, and telephone log functionalities to name just a few. These document-centric functionalities can be defined by the needs of the user and the creativity of third party software designers that can design extensions for the single window application, as will be explored in the section entitled "Extensible Functionalities" below. In the illustrated example, all of the documents are defined or created in HTML, although other mechanisms of defining or creating document can of course be used.

## **Document Modes**

The document-centric functionalities are characterized, in part in the present example, by the different types of documents that can be encountered. For example, there are "free-form"-type documents such as a word processing document, "form"-type documents such as a "personal contact" form in the contacts functionality where the user can insert certain information in certain blocks, and documents that fall somewhere in the middle between free-form and

1 form documents, e.g. an email message where there is a form at the top for the  
2 addressee and subject, and an open area underneath for text. In the described  
3 embodiment, multiple document modes are provided and essentially constrain the  
4 actions that a user can take relative to particular documents. Specifically, in the  
5 present example there are two document modes that a document can assume—a  
6 browse mode and an edit mode.

7 The document modes are useful because command sets that are displayed  
8 in the command area(s) 304 (Fig. 3) may change drastically depending on whether  
9 the user is simply viewing a browse-mode document or authoring an edit-mode  
10 document. In the present example, the document that is visible in the display area  
11 306 is always in a particular mode. For example, when a document is in browse  
12 mode, the user cannot change the underlying HTML that makes up the document.  
13 As an example, consider a case where a user navigates to a web page. The web  
14 page is displayed in the display area in a browse mode so that the user cannot  
15 change the underlying HTML that makes up the web page. (The page may have  
16 text areas and other things that can take user input, though.) When, however, a  
17 document is in edit mode, the user can change the content (i.e. HTML) that makes  
18 up the document. The amount that can be changed depends on the type of  
19 document provided: free-form documents allow more parts of the HTML to be  
20 edited than form items, for example. Consider that a user might navigate to their  
21 word processing functionality to a document collection and pull up a document.  
22 This document might be pulled up in browse mode so that the user can read it. If,  
23 however, the user wishes to manipulate the document, they can convert it to edit  
24 mode to manipulate the content of the document. To convert a document to edit  
25 mode, the user need simply click an "edit" button which then enables the user to

1 edit the document. There is policy that can be defined for when a document  
2 should be in edit or browse mode. For example, if the author places a document in  
3 edit mode during a session and then navigates to another document without  
4 finishing editing activities, then the edit-mode document will be placed in edit  
5 mode the next time the author visits the document. In the described embodiment,  
6 "new" documents that are created by the user are in edit mode by default. This  
7 allows the user to modify the document. In addition, the user can change the  
8 mode of a document from edit to browse, e.g. by signifying completion of the  
9 document. For example, for a document in the form of an email message, the  
10 mode change from edit to browse can be affected by sending the email message.  
11 For a document in the form of a contact, the mode change from edit to browse can  
12 be affected by saving the contact.

13 There are some navigation model issues that arise from having different  
14 document modes. Consider, for example, a user that navigates the single window  
15 to a document that is in browse mode and then converts the document to edit  
16 mode. Assume now that the user begins to edit the document but before they are  
17 through editing the document, they receive an email message from a friend. Using  
18 the navigation function, they click on the mail link to pull up their email box and  
19 then click on the new message. At this point, the navigation stack looks like this:

20  
21 Edit mode document→email box→new message  
22

23 Assume that when they are finished reading the new message, they wish to  
24 return to the document that they were editing. Rather than navigating back  
25 through the navigation stack using the "back" button, the user clicks on the



## Authoring

In the document-centric example, authors or users can both create new items and edit many existing items. In the illustrated example, the documents are defined in terms of HTML, although other formats can be used to define the documents. Authoring can take place in a freeform or form format. To author a new item, the user simply clicks a "new" button in the command area 304 (Fig. 3). The user can create items of different types from the same control (i.e. new email types, document types, contact types etc.). For example, a "New" button can provide a dropdown menu to allow the user to choose documents of different types. When a user creates a new document type, the navigation window is navigated to a new empty document of the corresponding type and the new item is entered into the navigation stack in edit mode by default. This new document remains in the navigation stack although provision can be made for maintaining links to the document even if it falls out of the navigation stack.

Fig. 10 is a flow diagram that describes steps in a document creation method in accordance with the described embodiment. The method is advantageously implemented in software. Step 1000 receives user input to author a new document type. The document type can be any suitable document type with exemplary document types including, without limitation, documents, email messages, calendar appointments, contacts and the like. Responsive to the user's input, step 1002 navigates the single window to a new empty document of a corresponding type. Thus, if the user specified a free-form document as the document type, the single window would be navigated to a free-form document; if the user specified an email message as the document type, the single window would be navigated to an empty email message form to be filled in by the user.

1 Step 1004 then makes an entry in the navigation model (e.g. the navigation stack)  
2 corresponding to the new document type. By default, the new document type is  
3 entered in the navigation stack in the edit mode so that the user can author the  
4 document.

5 Authors can also edit existing items, as indicated above. Typically, these  
6 items are items that they have worked with before. To edit an existing item, the  
7 user navigates their single window to the appropriate document and clicks on the  
8 "edit" button to convert the document from browse mode to edit mode. In the  
9 described embodiment, the navigation stack is manipulated as follows to  
10 accommodate the user's action. When the user navigates to a document of  
11 interest, an entry is made in the navigation stack that corresponds to the document  
12 in browse mode. After all, it was a browse mode document that the user pulled  
13 up. When the user clicks the "edit" button, the browse mode document is  
14 converted to an edit mode document in the navigation stack. One way of  
15 implementing this action is to perform a quick forward and delete operation. That  
16 is, when the document is converted into the edit mode, an edit mode entry for a  
17 copy of the browse mode document is made in the navigation stack and then the  
18 corresponding navigation stack entry for the browse document is deleted.  
19 Accordingly, this is but one more example of how the navigation model can be  
20 manipulated in accordance with the user's actions to preserve a consistent user  
21 context.

## 22 23 **Publishing**

24 When a user has completed a document, they can mark the document as  
25 complete by publishing it. Publication of a document means different things for



1 different types of documents. For example, a free-form document might be  
2 published to a central server; a contact might be published to the user's own  
3 section of the server; an email message is published when it is sent to the recipient.  
4 In the described embodiment, the functionalities in the single navigable window  
5 application are programmed so that they make sure the single navigable window  
6 understands each of the document types with which it is associated and how to  
7 specifically publish each of the documents. Thus a user, when finished working  
8 on a particular document, might click a "publish" button whereupon the software  
9 automatically knows how to publish the corresponding document. This is just one  
10 example of how an item might be published. Accordingly, the user need not know  
11 protocols or particular locations where the items are to be published.

12 Fig. 11 is a flow diagram that describes a publishing method in accordance  
13 with the described embodiment. The described method is advantageously  
14 implemented in software. Step 1100 creates or edits a document. This step is  
15 implemented by a user interacting with the single navigable window to either  
16 create a new document or edit an existing document as described above. Step  
17 1102 receives user input indicating that the user has completed the document. In  
18 the described example, this step is implemented responsive to a user clicking on a  
19 button such as a "publish" button. Responsive to the user providing this input,  
20 step 1104 automatically publishes the document based upon the document type.  
21 In this step, the software is programmed so that the publishing routine that is  
22 selected is specifically tied to the document type so that a user need not be  
23 concerned with any specific protocols or document locations. Rather, the software  
24 automatically publishes the document in a manner that is consistent with the  
25 document type.

## Viewing Collections

In the described embodiment, one of the advantageous features of the single navigable window application is that it can natively create (via the different built in functionalities), understand, and enable viewing of collections of documents of different types. For example, the application can allow the user to view any collection of documents that the user or others might have created based upon various properties of the documents such as document type or document author. To do this, a user simply navigates the single window to a collection of interest (such as the email box of Fig. 5, or a day on the calendar). The collection then appears in the display area 306 (Fig. 3) of the UI. Once the user has navigated to the particular collection, the software allows, in many cases, selection of a particular item (i.e. selection of a particular mail message in the email box) and navigation of the single window to that item. Thus, the selected item is viewed within the same window as was the particular collection. While at the various collections, the user can be presented with various context-sensitive tools to work within that collection. Examples of context-sensitive tools are described in the U.S. Patent Application entitled "Task Sensitive Methods And Systems For Displaying Command Sets" incorporated by reference above.

Accordingly, in this embodiment, the single navigable window application is able to navigate to and permit viewing of collections of different types of documents. This constitutes a noteworthy departure from conventional browser functionality which does not typically allow navigation and/or user interaction with different types of documents.

## Extensible Functionalities

In one embodiment, the functionalities that are provided by the single navigable window application are extensible meaning that the functionalities can be added and removed from the application. In this way, third party developers can provide various extensible functions that can be added or integrated into the single application. In this way, a user can add to their existing set of functionalities by simply selecting one or more functionalities of interest and incorporating the functionalities into the application. In addition, system administrators can add to the existing set of functionalities for a group of users by using a similar selection mechanism. These particular extensible functionalities are referred to as "extensions". Exemplary extensions and methods of providing extensions are described in U.S. Patent Applications entitled "Network-based Software Extensions", and "Architectures For And Methods Of Providing Network-based Software Extensions" incorporated by reference above.

## **Generic Single Navigable Window Application Platform**

One of the advantages of having the extensible functionalities as described above is that now, the single navigable window application can be provided as essentially a shell or software platform upon which these various functionalities can build. Consider for example Fig. 12. There, a single navigable window platform 1200 is shown. Platform 1200, in its most basic form, may have no functionalities whatsoever associated with it. Rather, it may only provide the infrastructure support that implements the single window navigation operability, e.g. software code that provides and manages a navigation model and navigation operations (such as clicking on a link to navigate a window to a particular

1 functionality). In this case, some or all of the functionalities could then be defined  
2 by third parties and added to the platform as appropriate. As an example,  
3 functionalities 1202, 1204, and 1206 have been added to the platform to impart a  
4 degree of functionality to the single navigable window application. These added  
5 functionalities might include a word processing functionality, an email  
6 functionality, and a calendar functionality. Other functionalities such as  
7 functionalities 1208, 1210, and 1212 have not been but could be added.

8 Delivery of the functionalities or extensions as they have been referred to  
9 above, can be in any suitable way. For example, an individual user might  
10 physically insert a CD carrying software code for a particular functionality into  
11 their computer and load the software. The functionality would then be added to  
12 the platform for the user to interact with. Alternately, the functionalities or  
13 extensions can be delivered over a network such as the Internet. An example of  
14 how this can be done is described in the Application entitled "Network-based  
15 Software Extensions", incorporated by reference above.

### 16 17 Subscription model

18 In accordance with one embodiment, the single navigable window  
19 application and its various functionalities are packaged and provided to consumers  
20 as a service to which they can subscribe for a fee. As an example model, consider  
21 Fig. 13. There, multiple clients are shown each of which contains a platform such  
22 as the one described in connection with Fig. 12. One or more servers are  
23 configured to provide various functionalities or extensions for delivery over the  
24 Web. The various functionalities or extensions can be provided by independent  
25 software vendors and can be written so that they plug directly into the software

1 platform that provides the single navigable window functionality. Mechanisms for  
2 plugging software extensions into a software platform will be understood by those  
3 of skill in the art and are not described in detail here.

4 As an example, consider the case of a small company that employs a  
5 computing system to track inventory and perform various office tasks such as  
6 word processing, email, accounting (i.e. payroll) and the like. Today, this  
7 company might employ several different software applications that provide all of  
8 the functionality that is needed for the organization. With that approach, the  
9 information technology costs can be very large, particularly for a small company.  
10 In the present model, however, a Web-based service ensures that each customer,  
11 e.g. each company, can access the various functionalities they need for a  
12 subscription fee. The functionalities are plugged into and integrated into a single  
13 navigable window application that greatly facilitates the customer's computing  
14 experience. It will be appreciated that, although each illustrated client in Fig. 13 is  
15 shown as having a platform to which the functionalities can be added, individual  
16 platforms for particular customers might be hosted by a separate server so that a  
17 customer need only log into the Web to have all of the subscribed functionalities  
18 exposed to them.

## 19 20 Conclusion

21 The embodiments described above provide improved methods and systems  
22 for creating and using information in a computing environment. The inventive  
23 methods and systems address and provide solutions to user issues relating to  
24 computing in a multi-window environment. The described methods and systems  
25 provide a single navigable window that can be used by a user to navigate to and

1 between multiple different functionalities that are provided by a single application  
2 program. The functionalities enable the user to complete different tasks, and the  
3 single window enables the user to navigate between functionalities, and hence  
4 tasks, in a seamless manner. By having only one window, the user is relieved of  
5 the duties of managing multiple windows. By having all of the functionalities  
6 presented within a single application, the user is provided with a highly integrated  
7 software product that greatly improves the user's computing experience.

8 The described novel use of a navigation model that manages the user's  
9 navigation activities to and between the different functionalities ensures that the  
10 user's navigation experience bears an accurate logical relationship with the user's  
11 various activities. The provided navigation instrumentalities enable the user to  
12 navigate among the different functionalities in a quick and efficient manner.

13 In the software platform embodiment, the single application is extensible to  
14 receive and incorporate different functionalities that are provided as software  
15 modules that can be sent over a network such as the Internet. The extensible  
16 software platform provides a basis to offer a subscriber or fee-based service where  
17 different subscribers can, for a fee, access different functionalities via a network  
18 such as the Internet.

19 Although the invention has been described in language specific to structural  
20 features and/or methodological steps, it is to be understood that the invention  
21 defined in the appended claims is not necessarily limited to the specific features or  
22 steps described. Rather, the specific features and steps are disclosed as preferred  
23 forms of implementing the claimed invention.